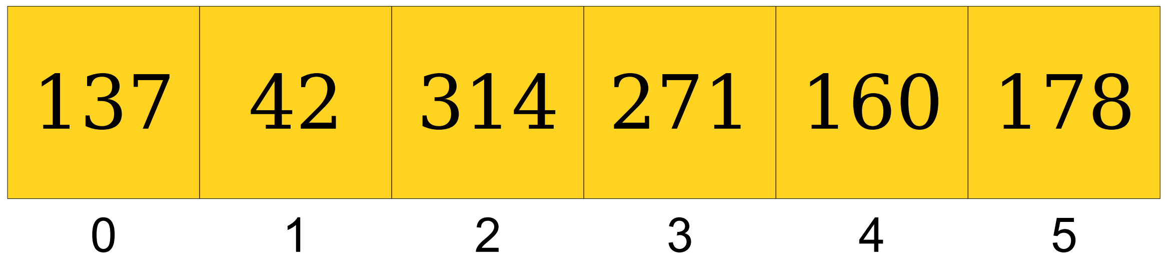
Arrays

A Different Way to Store Data

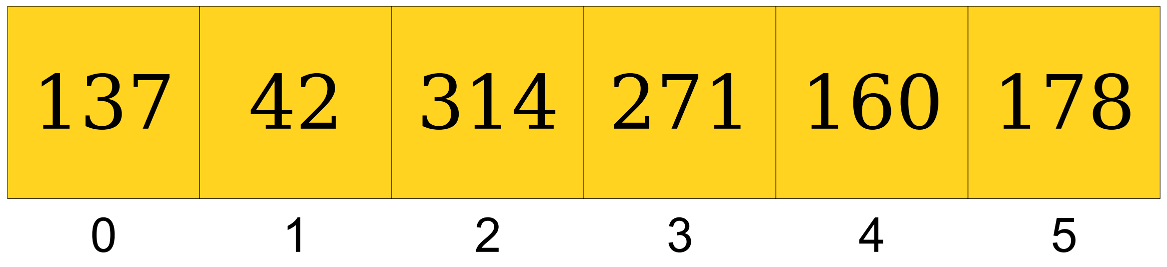
- On Monday, we saw the `ArrayList` as a way to store lots of data.
  - Lines of text.
  - US cities!
- Java also supports a concept called the ***array*** that can used to store lots of data.

Recapping `ArrayList`



- An `ArrayList` stores a sequence of multiple objects.
  - Can access objects by index by calling **`get`**.
- All stored objects have the same type.
  - You get to choose the type!
- Must store objects; primitive types not allowed.
- Can grow as long as it needs.

Introducing Arrays



- An array stores a sequence of multiple objects.
  - Can access objects by index using square brackets (more on that soon).
- All stored objects have the same type.
  - You get to choose the type!
- Can store *any* type, even primitive types.
- Size is fixed; cannot grow once created.

Basic Array Operations

- To create a new array, specify the type of the array and the size in the call to `new`:

```
Type[] arr = new Type[size]
```

- To access an element of the array, use the square brackets to choose the index:

```
arr[index]
```

- To read the length of an array, you can read the `length` field (without parentheses):

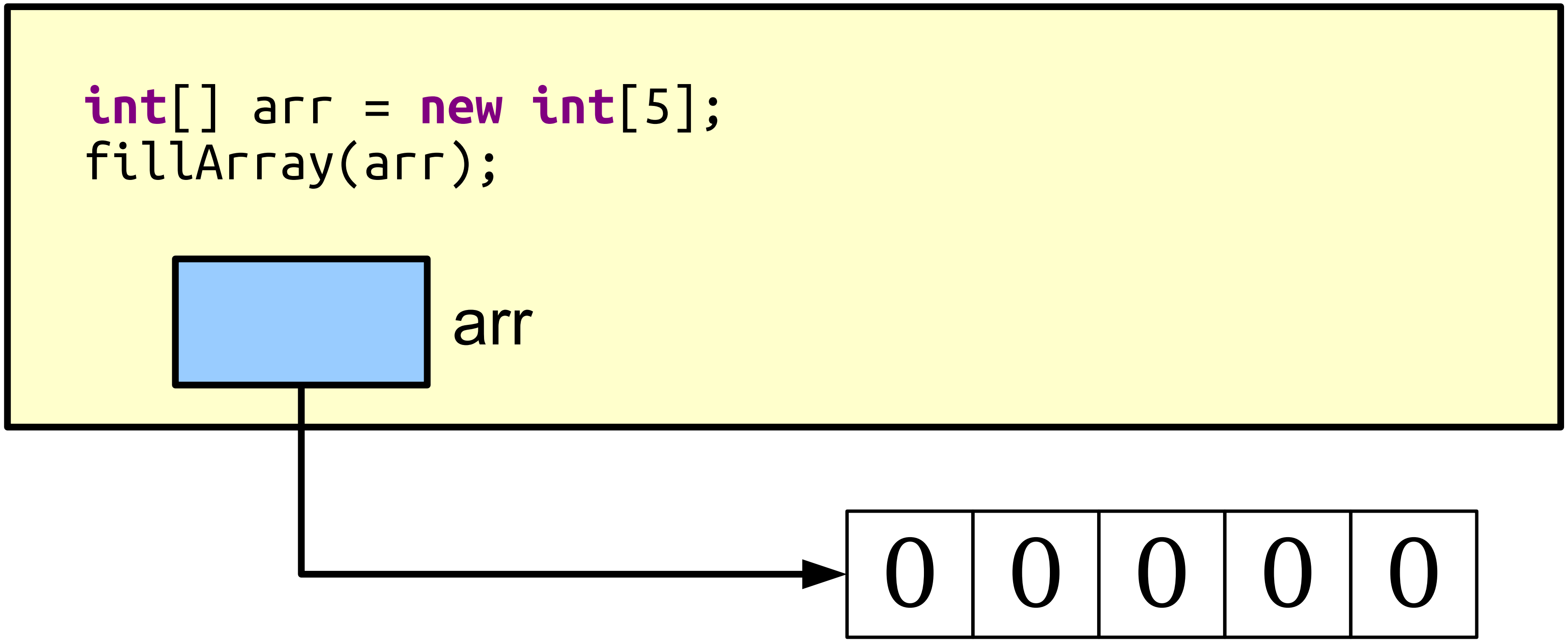
```
arr.length
```

Default Values in Arrays

- Because arrays have a fixed size, when declaring an array, all values in that array will initially be set to a default value:
  - `int`, `double`, etc. default to 0,
  - `boolean` defaults to `false`, and
  - Objects default to `null`.

Arrays as Parameters

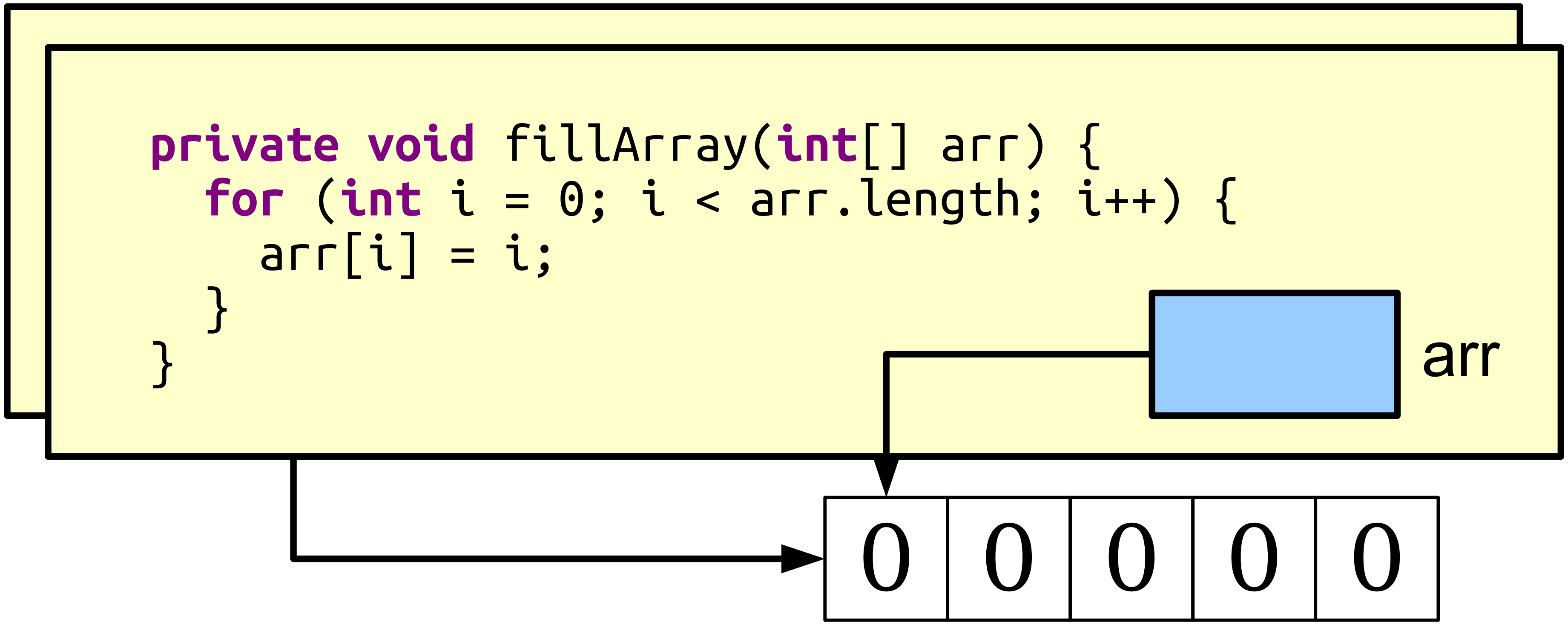
- Arrays are objects, so they obey the normal rules for passing objects into methods.
- The elements of an array can be modified inside of a method.



```
int[] arr = new int[5];
fillArray(arr);
```

Arrays as Parameters

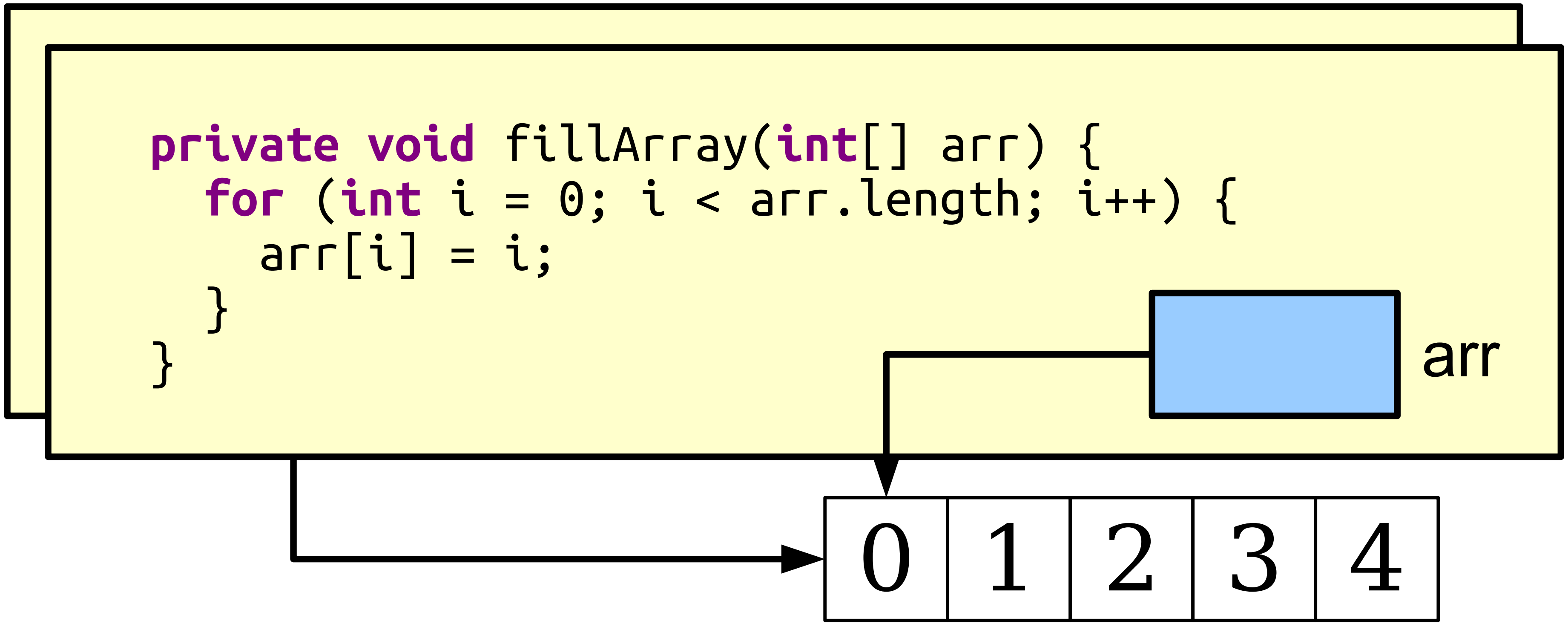
- Arrays are objects, so they obey the normal rules for passing objects into methods.
- The elements of an array can be modified inside of a method.



```
private void fillArray(int[] arr) {
  for (int i = 0; i < arr.length; i++) {
    arr[i] = i;
  }
}
```

Arrays as Parameters

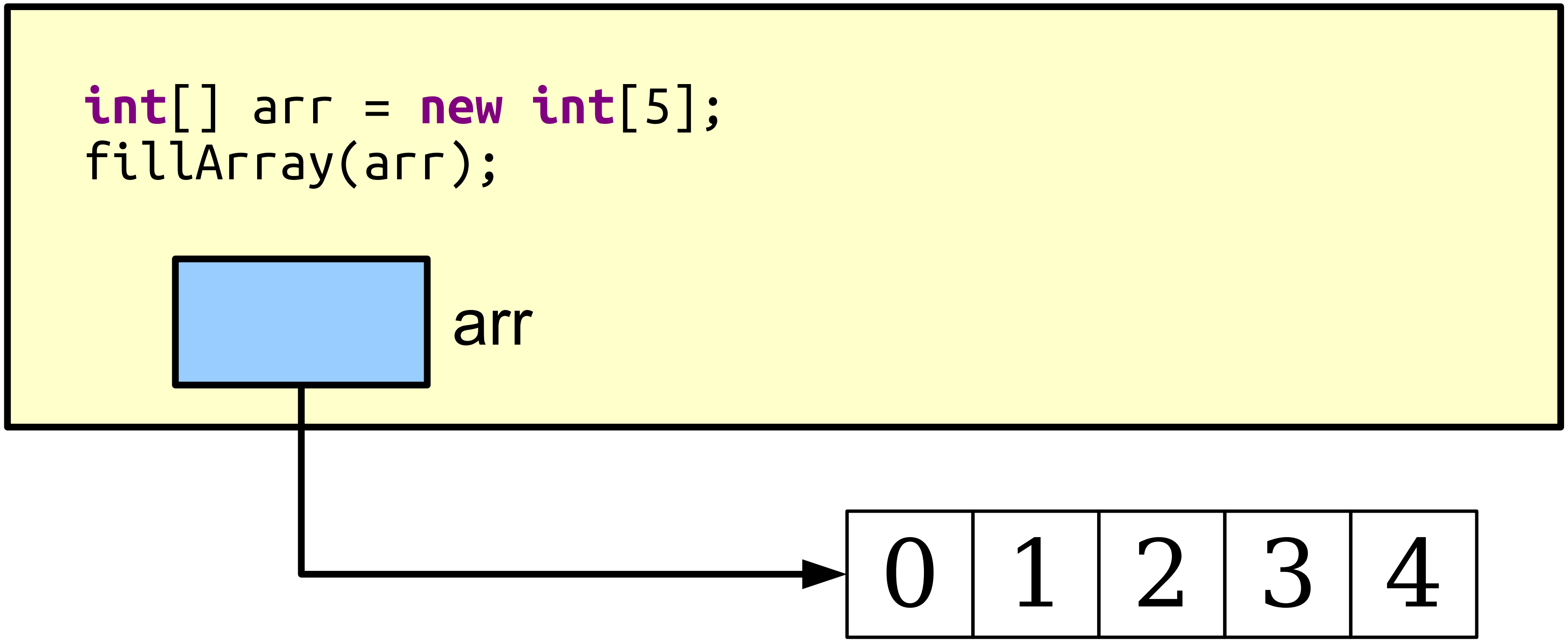
- Arrays are objects, so they obey the normal rules for passing objects into methods.
- The elements of an array can be modified inside of a method.



```
private void fillArray(int[] arr) {
  for (int i = 0; i < arr.length; i++) {
    arr[i] = i;
  }
}
```

Arrays as Parameters

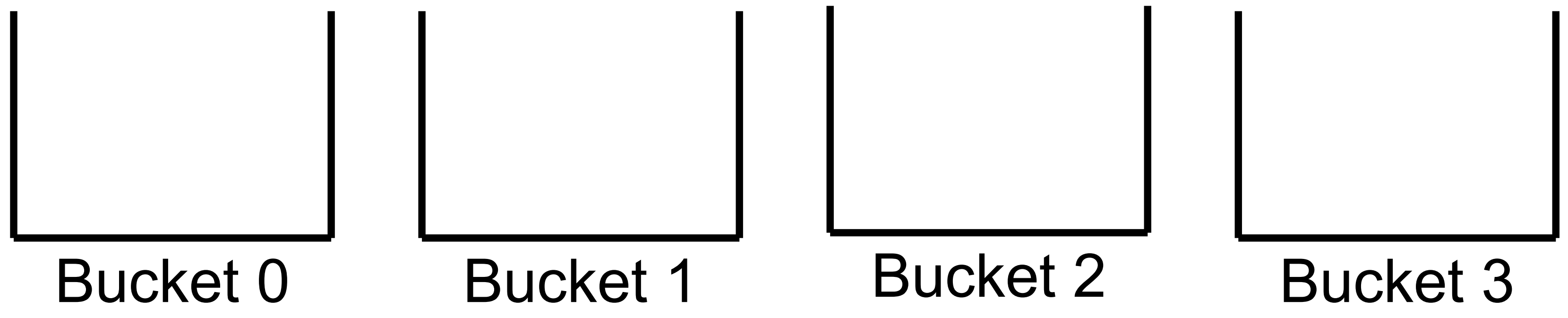
- Arrays are objects, so they obey the normal rules for passing objects into methods.
- The elements of an array can be modified inside of a method.



```
int[] arr = new int[5];
fillArray(arr);
```

Why Arrays?

- Arrays are excellent for representing a fixed-size list of ***buckets***.
- We can store values in the appropriate bucket by looking up the bucket by index.



How many people need to be in a room before two of them will share a birthday?

The Birthday Paradox

- In a room of 23 people, there is a 50% chance that two of them have the same birthday.
- More generally, if you have an *n*-sided die, you only need to roll it around $\sqrt{2n}$ times before you have a 50% chance of getting the same outcome twice.

Fun programming exercise: How many people do you need, on average, for three people to share a birthday?

Time-Out for Announcements!

Assignment 5

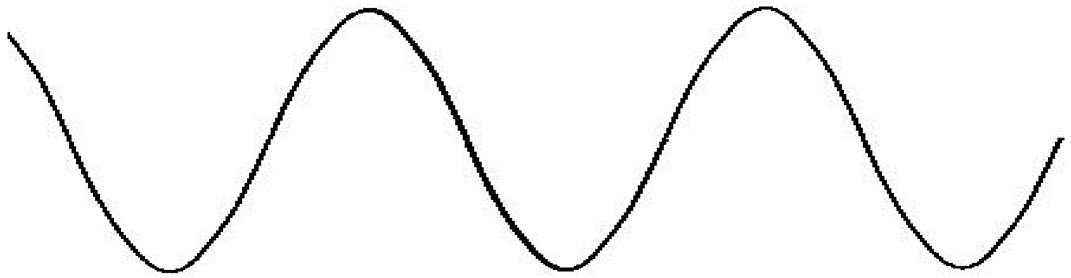
- Assignment 5 is due next Wednesday at 3:15PM.
- Recommendations:
    - Complete the syllable counting and algorism parts of the assignment by Friday. Test them extensively!
- Questions? Feel free to stop by the LaIR.

Back to CS106A!

Sound Processing

The Physics of Sound

- Sound is a wave that propagates through the air.



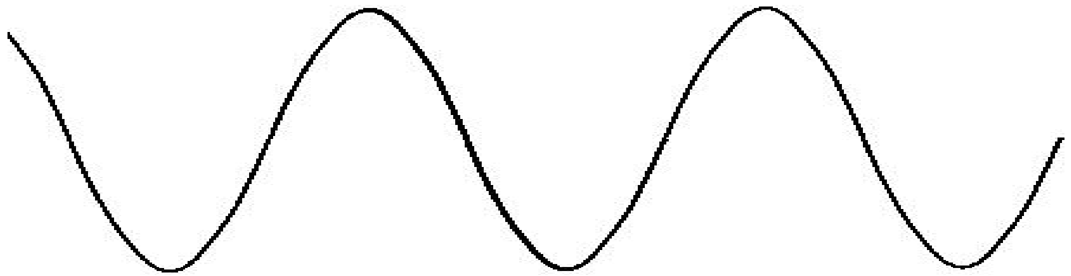
- The ***frequency*** of the wave is how closely packed together the peaks are.
  - Corresponds to ***pitch***.
- The ***amplitude*** of the wave is how tall the peaks are.
  - Corresponds to ***loudness***.

Representing Sound

- The computer can represent a sound by storing the sound wave.

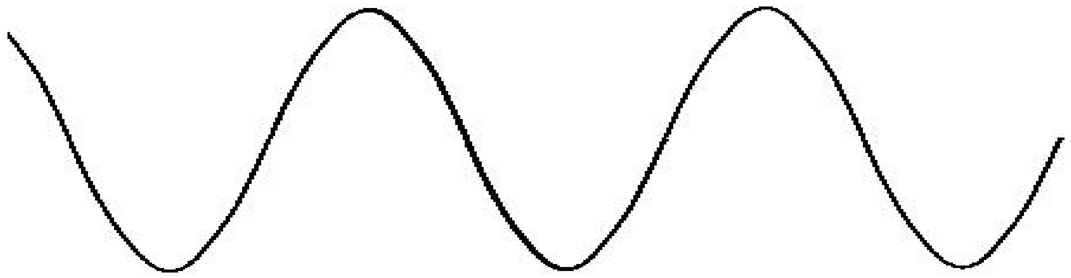
Representing Sound

- The computer can represent a sound by storing the sound wave.



Representing Sound

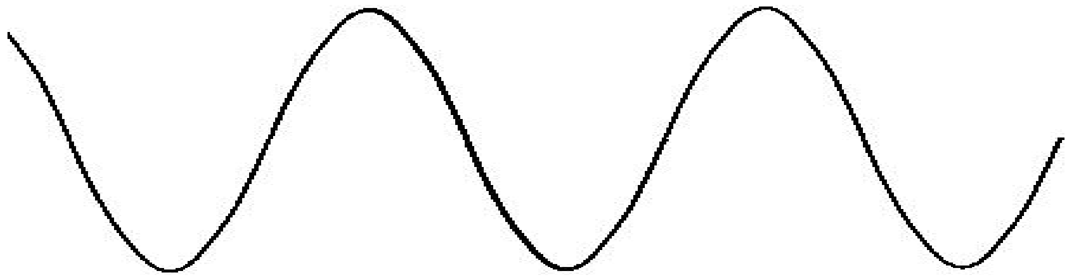
- The computer can represent a sound by storing the sound wave.



- Unfortunately, the wave is continuous, so the computer cannot store it completely.

Representing Sound

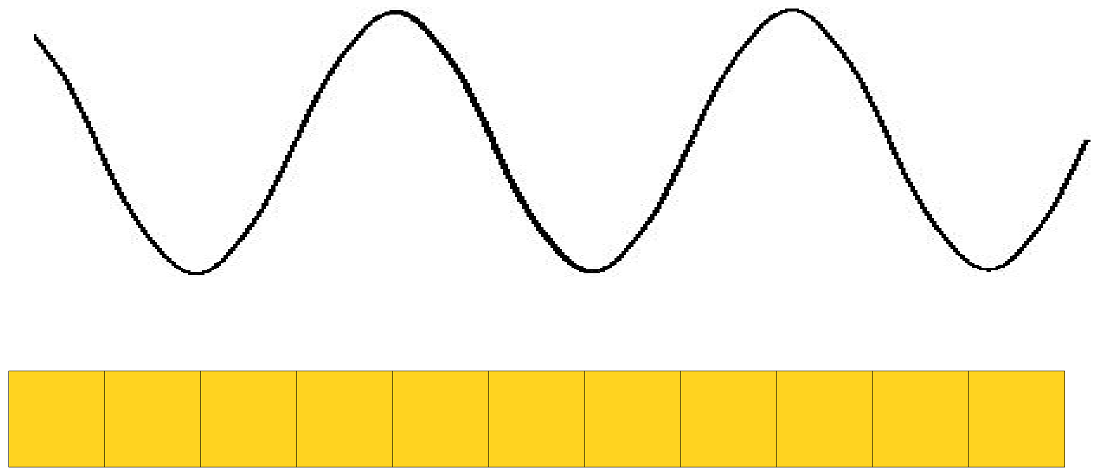
- The computer can represent a sound by storing the sound wave.



- Unfortunately, the wave is continuous, so the computer cannot store it completely.
- ***Idea:*** Sample points from the sound wave and store those instead.

Representing Sound

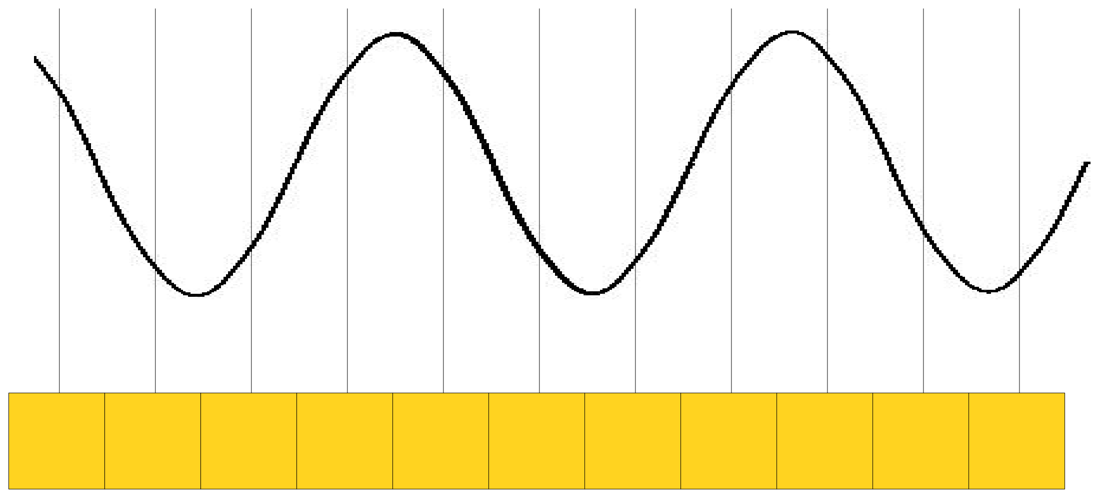
- The computer can represent a sound by storing the sound wave.



- Unfortunately, the wave is continuous, so the computer cannot store it completely.
- ***Idea:*** Sample points from the sound wave and store those instead.

Representing Sound

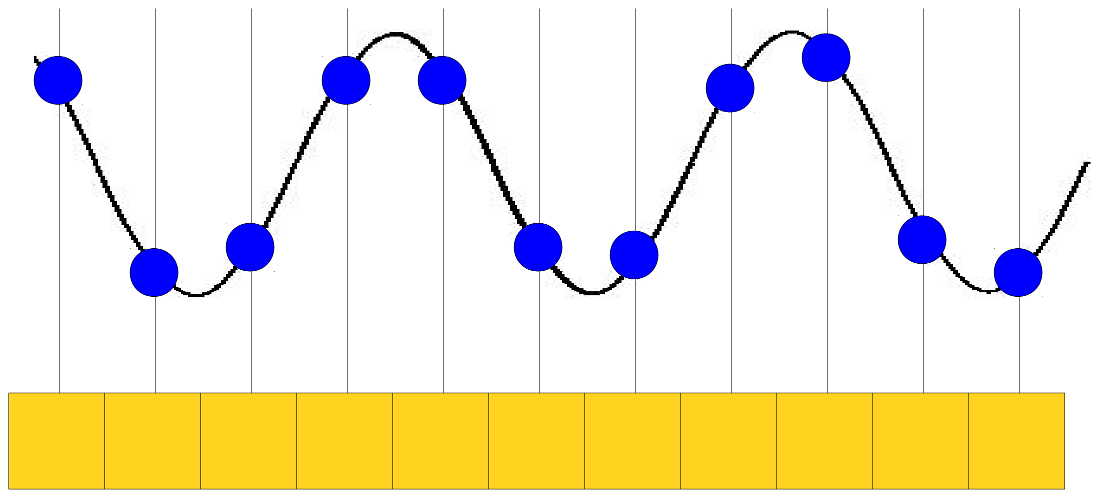
- The computer can represent a sound by storing the sound wave.



- Unfortunately, the wave is continuous, so the computer cannot store it completely.
- ***Idea:*** Sample points from the sound wave and store those instead.

Representing Sound

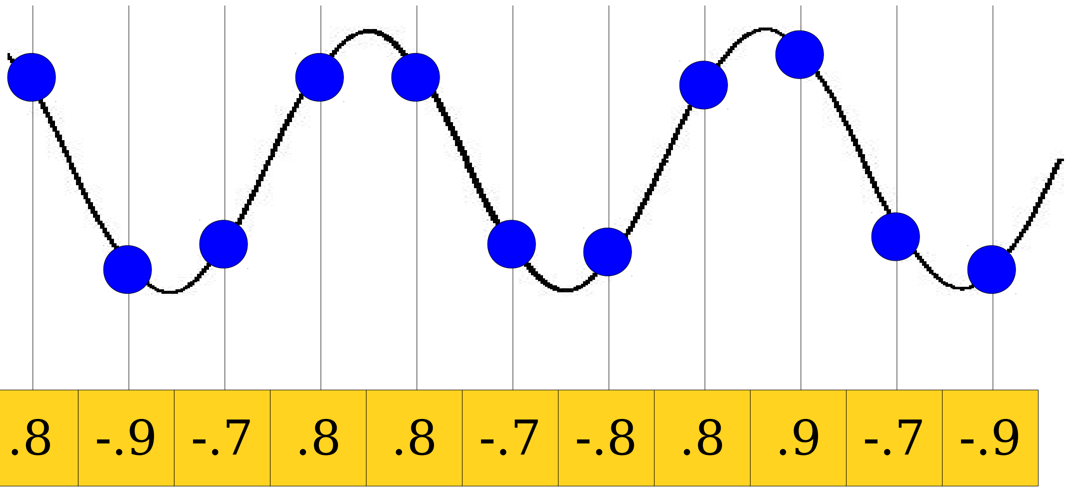
- The computer can represent a sound by storing the sound wave.



- Unfortunately, the wave is continuous, so the computer cannot store it completely.
- ***Idea:*** Sample points from the sound wave and store those instead.

Representing Sound

- The computer can represent a sound by storing the sound wave.



- Unfortunately, the wave is continuous, so the computer cannot store it completely.
- ***Idea:*** Sample points from the sound wave and store those instead.

The Sampling Rate

- The ***sampling rate*** of a sound clip is the frequency at which the wave's intensity is recorded.
  - Measured in hertz (Hz).
- Example: If sampling rate is 44,100Hz, there are 44,100 samples per second.
- High sampling rate makes for better sound.
- Low sampling rate uses less storage space.

Playing Sound

- Today, we'll use Princeton's `StdAudio` class to play sounds.
- Each sound clip is represented as a `double[]`, where each entry is between -1 and +1.
- We can play the sound by calling

```
StdAudio.play(soundClip)
```

Loading Sounds

- You can load .wav files with the appropriate sampling rate by calling

```
double[] clip = StdAudio.read(filename);
```

- Once you have that sound clip, you can do whatever you'd like with it!